

浙江大学



Matlab 图像处理编程实践 第一次大作业报告

课程名称:	Matlab图像处理
姓 名:	王晨雨
学 院:	计算机科学与技术学院
系:	计算机科学与技术
专 业:	计算机科学与技术
学 号:	3200102324

2022 年 7 月 3 日

目 录

1. 实验任务简介.....	3
2. 程序框架与技术细节.....	3
2.1 程序框架展示.....	3
2.2 实现细节.....	5
3. 程序运行示例.....	5
4. 实验结果分析.....	5
4.1 实验结果展示.....	5
4.2 结果分析	7
4.3 其他方法的对比.....	7
5. 总结与思考.....	8

1. 实验任务简介

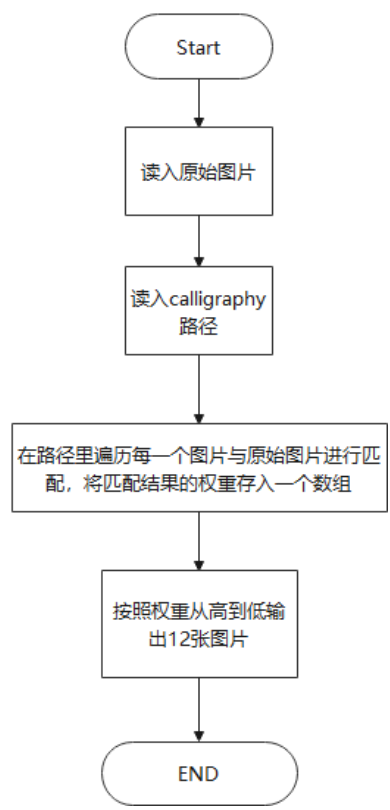
本次实验要求我们用 matlab 做一个相似书法字的检索工具。给定一个图片集，根据一些算法来实现在图片集中检索与输入的图片相似的书法图片。
比如输入为蔡襄_01 的春，输出匹配度最高的 12 张书法图为：



2. 程序框架与技术细节

2.1 程序框架展示

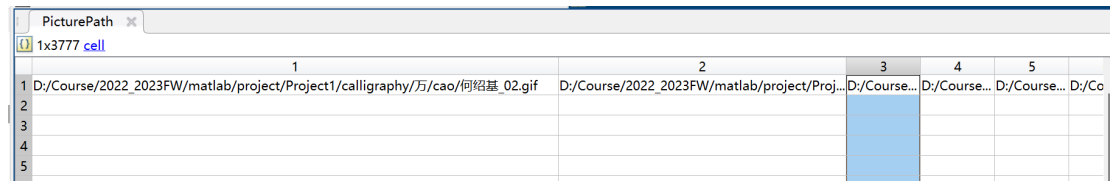
总体程序框架（流程图展示）：



FindDir.m:

```
function [PicturePath] = FindDir(RootDir)
```

该函数通过输入的 calligraphy 的根路径 RootDir，找到所有与原图像匹配的图像路径存入 PicturePath，比如输入我电脑中的 calligraphy 的根路径时，PicturePath 中存放 1x3777 个 cell，里面存的是每个待匹配图片的路径。将 PicturePath 返回上层 main.m。



	1	2	3	4	5
1	D:/Course/2022_2023FW/matlab/project/Project1/calligraphy/万/cao/何绍基_02.gif	D:/Course/2022_2023FW/matlab/project/Proj...	D:/Course...	D:/Course...	D:/Course... D:/Co
2					
3					
4					
5					

检索 PicturePath 中的每一个图像:

Cut 函数将每个图象分成 3*3 的块，转换为 1*9 的向量。

Similar 函数对比两个 1*9 的向量的相似度。注意如果是 rgb 图像要先转换为灰度图像再转换为二值图像。

```
for i = 1:n
    if strcmp(iminfo(PicturePath{i}).ColorType, 'indexed') ||
        strcmp(iminfo(PicturePath{i}).ColorType, 'grayscale')
        compare1 = Cut(binA);
        compare2 = Cut(imbinarize(imread(PicturePath{i})));
        result(i) = Similar(compare1, compare2);
    elseif strcmp(iminfo(PicturePath{i}).ColorType, 'truecolor')
        compare1 = Cut(binA);
        tmpGray = rgb2gray(imread(PicturePath{i}));
        compare2 = Cut(imbinarize(tmpGray));
        result(i) = Similar(compare1, compare2);
    end
end
```

Cut.m:

```
function [result] = Cut(binPicture)
```

先用 imresize 函数将图片压缩为[370,370]的格式，然后切分为 3*3 的块，计算每个块为 0 的值，存入 result 中。

```
binPicture = imresize(binPicture, [370, 370]);
result = zeros(1, 9); % cut into 3x3 pieces
```

Similar.m:

```
function [result] = Similar(compare1, compare2)
```

比较两个 1*9 向量，compare1 和 compare2 的相似度，通过公式：

$$similarity = \cos(\theta) = \frac{A * B}{||A|| ||B||} = \frac{\sum_{i=1}^n A_i * B_i}{\sqrt{\sum_{i=1}^n A_i^2} * \sqrt{\sum_{i=1}^n B_i^2}}$$

来计算相似度，并且把相似度存入 result 中返回上层 main.m。

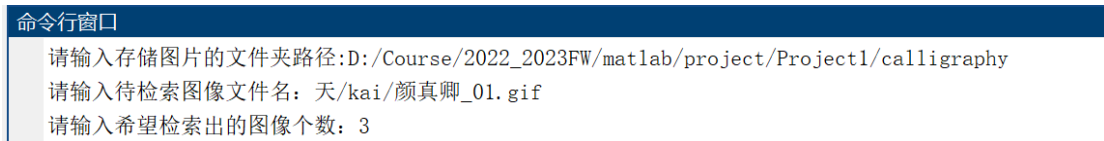
最后上层 main.m 中将相似度排序后由高到低输出相似度最高的图像。

2.2 实现细节：

1. 注意统一图片格式，把图片统一通过 `imbinarize` 函数转换为二值图像，再进行匹配。
2. 输出结果的时候也都输出二值图像。

3. 程序运行示例

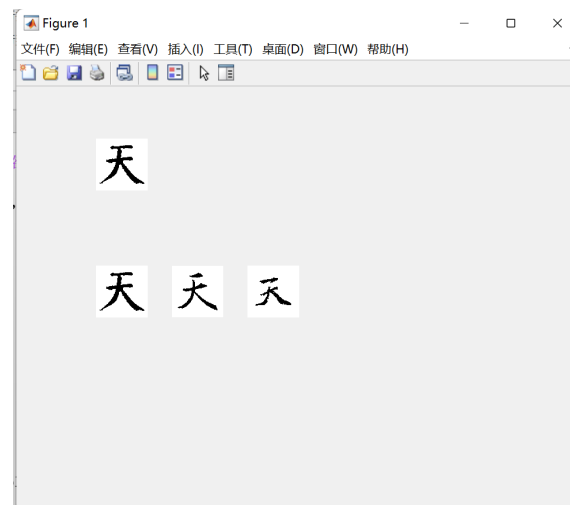
根据提示语输入：



注意：待检索图像文件名请从 calligraphy 下的那一层路径开始输入。如上图所示的“天/kai/颜真卿_01.gif”

待检索图像个数请在 2-12 范围内，因为我的子图大小只设置了可以放下 12 个图大小的位置。

比如上面的输入，输出结果即为：



由于嵌套循环语句较多，整个程序跑一次可能需要将近一分钟的时间，请耐心等待。

4. 实验结果分析

4.1 实验结果展示

以下为我的程序测试出测试用例（都以输出 6 个结果图像为例）：

输入图像	输入图像	测试结果	准确率
天/kai/ 颜真卿_01.gif			83%
惟/kai/ 高贞碑_01.gif			83%
日 /xing/ 索靖_01.gif			33%
我/kai/ 颜真卿_01.gif			100%
声 /xing/ 褚遂良_04.gif			33%
物/li/ 樊敏碑_01.gif			50%
仰 /xing/ 王羲之_01.gif			33%
枝/cao/ 智永_04.gif			83%
重/kai/ 欧阳询_01.gif			67%
者 /qita/ 欧阳询_02.gif			33%
知 /xing/ 冯承素_01.gif			67%
众/kai/ 褚遂良_01.gif			33%

字 /xing/ 米 芾_01. gif			50%
学 /kai/ 颜真 卿_02. gif			67%
物 /xing/ 王 羲之_01. gif			33%

4.2 结果分析：

书法字检索的正确率，主要取决于以下两个因素：

1. **书法字图像自身的性质。**书法字的字体、繁简、形状、结构相似字的干扰、图像放缩后字在整个图像中的位置与尺寸等等因素，都会影响其九维向量与自身性质的联系紧密性。多次测试表明，字形简洁或结构特征明显的书法字，检索正确率较高；笔画紧凑繁冗或有结构相似的字（如“字”与“学”）的书法字，检索正确率较低。
楷书相比其他书法字体来说检测正确率高一点。以及相同的书法家更容易检测出自己写的同一字体的另外版本（比如颜真卿的楷书“学”）。
2. **构造向量时的分割细度。**本程序将得到的二值图像分为九块方形区域。若分割更加精细（如分为 64 块得 64 维向量），则检索正确率还会有所提高。

出现的问题：

在其他字体中，有些篆刻或者拓本的字体，转化为二值图像后与普通的白纸黑字是相反的，比如结果测试中的“者/qita/欧阳询_02. gif”，这会带来检索中的问题。

4.3 其他方法的对比：

当今有三种常用的图像检索方法：

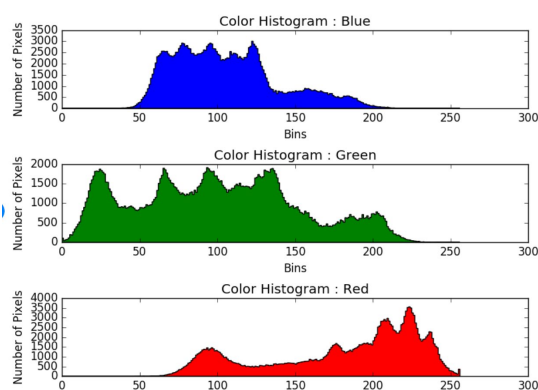
1. 感知哈希算法（Perceptual Hash Algorithm）
 - a) 感知哈希算法作用是对每张图片生成一个“指纹”（Fingerprint）¹字符串，然后比较不同图片的指纹。指纹越接近，就说明图片越相似。
 - b) 感知哈希算法生成指纹的方法为，将图像缩小到 64 个像素后灰度化后计算这 64 个像素点的平均值，将每个像素点与平均值比较，大于等于平均值记为 1，小于为 0，然后将比较结果组合在一起构成一个 64 位的二进制串，转换为 16 进制的 16 位数；这就是哈希值即“指纹”。

¹ Wikipedia contributors. (2022, June 9). Perceptual hashing. In Wikipedia, The Free Encyclopedia. Retrieved 01:10, July 6, 2022, from https://en.wikipedia.org/w/index.php?title=Perceptual_hashing&oldid=1092315030

- c) 得到指纹以后, 就可以对比不同的图片, 看看 64 位中有多少位是不一样的。如果不相同的数据位不超过 5, 就说明两张图片很相似; 如果大于 10, 就说明这是两张不同的图片。
- d) 感知哈希算法的特点是简单快速, 不受图片缩放的影响。最佳用途是根据缩略图找到原图。

2. 颜色分布法

- a) 将图片生成颜色直方图, 即分别绘制 R、G、B 的分布图。



(图 1) Color Histogram²

- b) 具体实现过程中, 可以将图片分区处理, 统计每个区域包含的像素数量, 组成一个向量, 即图片的特征值。
 - c) 最后, 根据图片的特征值比较图片的相似度。
- ## 3. 内容特征法
- a) 即比较图片内容的相似性。把缩小后的 RGB 图片通过某一个阈值二值化成黑白图片。(关键是阈值的选取, 比如大津法等)
 - b) 二值化后通过与颜色分布法一样的原理得到图片的特征矩阵后比较。

5. 总结与思考

通过本次 project, 我理解了一些图片在内存中的表示方式以及图片特征。我是将一张图片分成 3*3 的块来处理的, 从准确率来看并不十分理想。为此我有以下的思考:

1. 这种方法只能检索出大小和内容几乎一模一样的图片, 比如把同一个书法字体缩小后再放到原来的背景下与原图进行检索, 效果很差。
2. 可能单凭图像处理和计算像素特征值的方法局限性很大, 所以我认为可以在提取图像像素特征的同时也要识别图像内容, 根据图像内容做匹配效果会更好, 不过这也涉及到了人工智能的范畴。

² Li, Jung-Shian & Liu, I-Hsien & Tsai, Chin-Jui & Su, Zhi-Yuan & Li, Chu-Fen & Liu, Chuan-Gang. (2020). Secure Content-Based Image Retrieval in the Cloud With Key Confidentiality. IEEE Access. PP. 1-1. 10.1109/ACCESS.2020.3003928.